CONCURRENT SHARED OBJECT IMPLEMENTED USING A LINKED LIST WITH AMORTIZED NODE ALLOCATION

ABSTRACT OF THE DISCLOSURE

[1131] The Hat Trick deque requires only a single DCAS for most pushes and pops. The left and right ends do not interfere with each other until there is one or fewer items in the queue, and then a DCAS adjudicates between competing pops. By choosing a granularity greater than a single node, the user can amortize the costs of adding additional storage over multiple push (and pop) operations that employ the added storage. A suitable removal strategy can provide similar amortization advantages. The technique of leaving spare nodes linked in the structure allows an indefinite number of pushes and pops at a given deque end to proceed without the need to invoke memory allocation or reclamation so long as the difference between the number of pushes and the number of pops remains within given bounds. Both garbage collection dependent and explicit reclamation implementations are described.